Implementation Blueprint for a Promptable Image-Segmentation Service

Production-style reference: Dockerised REST API + CLI Client

Executive Summary

This report specifies a concrete, production-style (“Cisco-style”: clean structure, explicit governance, repeatable CI/CD, compliance-first data handling) implementation plan for a repository that hosts a promptable image-segmentation service with (a) a Dockerised REST API and (b) a small CLI client.

The emphasis is on: (1) public redistribution safety — no raw datasets, no ambiguous/proprietary weights in the repository; (2) repeatability — pinned dependencies, deterministic modes, container pinning; and (3) operational maturity — CI, releases, monitoring hooks, security policies.

Key Conservative Design Choices

• Model artefacts are **never committed to the repository**; instead they are fetched at runtime and verified via checksum. This avoids hard file limits (100 MiB blocks) and keeps the repo small and forkable.

• Model artefacts are distributed via versioned release assets by default, supporting large binaries without repo bloat.

• Datasets are consumed only via download scripts; the repo contains dataset 'cards' and attribution files, and validation scripts, but never stores raw images. This is essential because several popular segmentation datasets have licence constraints or provenance requirements.

• CI workflows are pinned, least-privilege, and supply-chain aware.

Repo Skeleton and Files

Target Repo Layout

A production-friendly layout that cleanly separates product code, MLOps scripts, infrastructure, monitoring assets, and documentation:

```
.
■■■ README.md
■■■ pyproject.toml
■■■ LICENSE
■■■ NOTICE
■■■ SECURITY.md
■■■ CONTRIBUTING.md
■■■ CODE_OF_CONDUCT.md
■■■ CHANGELOG.md
■■■ .gitignore
■■■ .dockerignore
■■■ .pre-commit-config.yaml
■■■ .editorconfig
■■■ Makefile
■■■ docker/
■   ■■■ Dockerfile
■   ■■■ entrypoint.sh
■■■ configs/
■   ■■■ service.yaml
■   ■■■ logging.yaml
■   ■■■ model_registry.json
■   ■■■ prometheus/
■       ■■■ prometheus.yml
■■■ data/
■   ■■■ README.md
■   ■■■ metadata/
■   ■   ■■■ datasets/
■   ■   ■   ■■■ openimages_v5_masks.md
■   ■   ■   ■■■ coco_2017_instances.md
■   ■   ■   ■■■ voc2012_segmentation.md
■   ■   ■   ■■■ davis2017.md
■   ■   ■   ■■■ youtube_vos.md
■   ■   ■■■ model_cards/
■   ■       ■■■ sam.md
■   ■       ■■■ sam2.md
■   ■       ■■■ mobilesam.md
■   ■■■ attribution/
■       ■■■ DATA_SOURCES.md
■       ■■■ THIRD_PARTY_NOTICES.md
■■■ scripts/
■   ■■■ datasets/
■   ■   ■■■ download_openimages_v5.sh
■   ■   ■■■ download_coco_2017.sh
■   ■   ■■■ download_voc2012.sh
■   ■   ■■■ download_davis2017.sh
■   ■   ■■■ download_youtube_vos.sh
■   ■■■ verify_checksums.py
■   ■■■ export_onnx.py
■   ■■■ package_release_assets.sh
■■■ src/
■   ■■■ son_segserve/
■       ■■■ __init__.py
■       ■■■ api/
■       ■   ■■■ app.py
■       ■   ■■■ routes.py
■       ■   ■■■ schemas.py
■       ■■■ cli/
■       ■   ■■■ main.py
■       ■■■ core/
■       ■   ■■■ config.py
■       ■   ■■■ logging.py
■       ■   ■■■ security.py
■       ■■■ model/
■       ■   ■■■ loader.py
■       ■   ■■■ registry.py
■       ■   ■■■ prompts.py
■       ■   ■■■ postprocess.py
■       ■■■ metrics/
■       ■   ■■■ prometheus.py
■       ■   ■■■ drift.py
■       ■■■ utils/
■           ■■■ hashing.py
■           ■■■ io.py
■■■ monitoring/
■   ■■■ grafana/
■   ■   ■■■ dashboard.json
■   ■■■ alerts/
■       ■■■ rules.yml
■■■ tests/
■   ■■■ unit/
■   ■■■ integration/
■   ■■■ regression/
■   ■■■ synthetic/
■■■ docs/
    ■■■ index.md
    ■■■ architecture.md
    ■■■ api.md
    ■■■ datasets.md
    ■■■ models.md
    ■■■ contributing.md
    ■■■ runbooks.md
```

Design rationale: Keep data as metadata + scripts only — this supports public redistribution while remaining legally conservative. Keep infra 'portable': Docker + optional compose + sample Prometheus/Grafana configs.

File-to-Purpose Mapping

File / Directory	Purpose	Notes / Acceptance Criteria
`README.md`	Product definition, quickstart, demo, governance entry points	Must contain 'no datasets/weights in git' note
`pyproject.toml`	Single source of truth for packaging, dependencies, scripts	Supports reproducible local + CI installs
`LICENSE`	Repo's code licence	Use a standard licence (Apache-2.0 or MIT recommended)
`NOTICE`	Attribution + legal notices	Include third-party notices + dataset attribution pointers
`SECURITY.md`	Vulnerability reporting policy	Include contact + response SLA
`CONTRIBUTING.md`	Contributor workflow and expectations	Recognisable standard location
`data/metadata/datasets/*.md`	Dataset cards for transparency and compliance	Include licence, splits, checksums, provenance notes
`scripts/datasets/*`	Repeatable dataset download + verification	Never commit dataset binaries
`configs/model_registry.json`	Machine-readable registry of model artefacts and checksums	Enables download-on-first-run verification
`monitoring/*`	Dashboards + scrape configs + alerts stub	Grafana dashboards are JSON models

Minimal Content Templates

README.md

~~~
# SON SegServe — Promptable Image Segmentation Service

A production-style reference implementation of a promptable segmentation service:
- REST API (containerised)
- CLI client (thin wrapper over the API)
- Model artefacts fetched at runtime + checksum-verified (no weights committed)

## Quickstart (Docker)
```bash
docker build -f docker/Dockerfile -t son-segserve:dev .
docker run --rm -p 8080:8080 -e SON_MODEL_ID=sam2_tiny son-segserve:dev
```

## Quickstart (CLI)
```bash
pip install -e ".[dev]"
son-segserve ping --api http://localhost:8080
son-segserve segment --api http://localhost:8080 --image ./demo.jpg \
  --prompt box:10,10,200,200
```

## Repo Principles
- Do not commit datasets or model weights to the repository.
- Use scripts/datasets/* to download open datasets locally.
- Use versioned release assets for model checkpoints + checksums.

## Docs
- Architecture: docs/architecture.md
- API: docs/api.md
- Datasets & licensing: docs/datasets.md
~~~

pyproject.toml

```
[project]
name = "son-segserve"
version = "0.1.0"
description = "Promptable image segmentation service (REST + CLI)"
requires-python = ">=3.10"
readme = "README.md"
license = {file = "LICENSE"}
authors = [{name = "SON Contributors"}]
dependencies = [
    "fastapi>=0.110",
    "uvicorn[standard]>=0.27",
    "pydantic>=2.6",
    "httpx>=0.27",
    "numpy>=1.26",
    "pillow>=10.0",
    "prometheus-client>=0.20",
    # torch / onnxruntime intentionally not pinned here:
    # prefer extra groups per backend/hardware
]

[project.optional-dependencies]
dev = [
    "pytest>=8.0",
    "pytest-cov>=5.0",
    "ruff>=0.4",
    "mypy>=1.8",
```

```
    "pre-commit>=4.0",
]
cpu = ["onnxruntime>=1.17"]
gpu = ["onnxruntime-gpu>=1.17"]

[project.scripts]
son-segserve = "son_segserve.cli.main:app"

[tool.ruff]
line-length = 100

[tool.pytest.ini_options]
addopts = "-q"
testpaths = ["tests"]
```

.gitignore (public-ML-safe defaults)

```
# Python
__pycache__/
*.pyc
.venv/
.pytest_cache/
.mypy_cache/
.ruff_cache/

# Local data / checkpoints (never commit)
data/raw/
data/processed/
data/cache/
models/
*.pt
*.pth
*.onnx
*.ckpt

# OS/IDE
.DS_Store
.idea/
.vscode/

# Build outputs
dist/
build/
*.egg-info/
```

.dockerignore

```
.git
.venv
__pycache__
*.pyc
data/raw
data/processed
models
dist
build
tests
docs
```

.pre-commit-config.yaml

```
repos:
  - repo: https://github.com/pre-commit/pre-commit-hooks
    rev: v5.0.0
    hooks:
      - id: end-of-file-fixer
      - id: trailing-whitespace
      - id: check-yaml
  - repo: https://github.com/astral-sh/ruff-pre-commit
    rev: v0.6.4
    hooks:
      - id: ruff
      - id: ruff-format
```

SECURITY.md

```
# Security Policy

## Supported Versions
Only the latest `main` branch and the latest tagged release are supported.

## Reporting a Vulnerability
Please do NOT open a public issue.
Email: security@example.org
Include:
  - version/tag/commit SHA
  - reproduction steps
  - impact assessment (if known)
We aim to acknowledge reports within 72 hours.
```

CONTRIBUTING.md

```
# Contributing

## Development Setup
python -m venv .venv && source .venv/bin/activate
pip install -e ".[dev]"
pre-commit install
pytest

## Pull Request Process
- One feature/fix per PR
- Add/adjust tests
- Update docs if behaviour changes
- Ensure ruff, mypy, and pytest pass locally

## Code Style
Run `pre-commit run -a` before pushing.
```

NOTICE

```
SON SegServe
Copyright (c) 2026 SON Contributors

This distribution includes third-party components; see
data/attribution/THIRD_PARTY_NOTICES.md.

Dataset and model licences are documented in data/metadata
and must be reviewed before commercial use.
```

CI/CD Workflows

The following YAML outlines reflect three non-negotiables:

- Least-privilege workflow permissions (job-level).
- Pin Actions to commit SHAs.
- Do not upload raw datasets; only upload build artefacts and small test fixtures.

CI Workflow (lint + unit + integration)

```yaml
name: CI
on:
  pull_request:
  push:
    branches: [main]

permissions:
  contents: read

jobs:
  test:
    runs-on: ubuntu-latest
    strategy:
      matrix:
        python: ["3.10", "3.11"]
    steps:
      - name: Checkout
        uses: actions/checkout@v5
      - name: Set up Python
        uses: actions/setup-python@v5
        with:
          python-version: ${{ matrix.python }}
      - name: Install
        run: |
          python -m pip install -U pip
          pip install -e ".[dev]"
      - name: Lint
        run: |
          ruff check .
          ruff format --check .
      - name: Type-check
        run: mypy src
      - name: Unit tests
        run: pytest -q --cov=son_segserve --cov-report=term-missing

  integration:
    runs-on: ubuntu-latest
    needs: [test]
    steps:
      - uses: actions/checkout@v5
      - run: docker build -f docker/Dockerfile -t son-segserve:ci .
      - run: |
          docker run -d --name api -p 8080:8080 son-segserve:ci
          python -m pip install -U httpx
          python scripts/ci_smoke.py --base-url http://localhost:8080
```

Docker Build + Push Workflow (Container Registry)

```yaml
name: Docker
on:
  release:
    types: [published]
```

```
env:
  REGISTRY: ghcr.io
  IMAGE_NAME: ${{ github.repository }}

jobs:
  build_push:
    runs-on: ubuntu-latest
    permissions:
      contents: read
      packages: write
      attestations: write
      id-token: write
    steps:
      - uses: actions/checkout@v5
      - name: Login to Container Registry
        uses: docker/login-action@<PINNED_SHA>
        with:
          registry: ${{ env.REGISTRY }}
          username: ${{ github.actor }}
          password: ${{ secrets.GITHUB_TOKEN }}
      - name: Extract metadata
        id: meta
        uses: docker/metadata-action@<PINNED_SHA>
        with:
          images: ${{ env.REGISTRY }}/${{ env.IMAGE_NAME }}
      - name: Build and push
        id: push
        uses: docker/build-push-action@<PINNED_SHA>
        with:
          context: .
          file: docker/Dockerfile
          push: true
          tags: ${{ steps.meta.outputs.tags }}
          labels: ${{ steps.meta.outputs.labels }}
      - name: Attest provenance
        uses: actions/attest-build-provenance@v3
        with:
          subject-name: ${{ env.REGISTRY }}/${{ env.IMAGE_NAME }}
          subject-digest: ${{ steps.push.outputs.digest }}
          push-to-registry: true
```

Note: Keep images lean via multi-stage builds and exclude caches to stay within layer limits.

Docs Deploy Workflow

```
name: Docs
on:
  push:
    branches: [main]

permissions:
  contents: read
  pages: write
  id-token: write

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v5
      - uses: actions/setup-python@v5
        with:
          python-version: "3.11"
      - run: |
          python -m pip install -U pip
          pip install -e ".[dev]"
          pip install mkdocs mkdocs-material
          mkdocs build
      - name: Upload Pages artifact
        uses: actions/upload-pages-artifact@v3
        with:
          path: site/
  deploy:
    needs: build
    runs-on: ubuntu-latest
    environment:
      name: github-pages
      url: ${{ steps.deployment.outputs.page_url }}
    steps:
      - name: Deploy
        id: deployment
        uses: actions/deploy-pages@v4
```

Release Workflow

```
name: Release
on:
  workflow_dispatch:
  push:
    tags: ["v*.*.*"]

permissions:
  contents: write

jobs:
  cut_release:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v5
      - name: Build model artefact bundle
        run: |
          bash scripts/package_release_assets.sh
          ls -lh dist/
      - name: Create release and upload assets
        env:
          GH_TOKEN: ${{ github.token }}
        run: |
          gh release create "${GITHUB_REF_NAME}" \
            --title "${GITHUB_REF_NAME}" \
            --notes "SON SegServe release ${GITHUB_REF_NAME}" \
            dist/*
```

Local Equivalent (Maintainer Runbook)

```
gh auth login
git tag v0.1.0 && git push origin v0.1.0
gh release create v0.1.0 dist/*
```

Model and Dataset Artefact Handling

Why You Must Not Commit Weights or Datasets

- Repositories block files over 100 MiB in normal pushes and recommend keeping repos small (ideal < 1 GB; strongly recommended < 5 GB).
- Git LFS has plan-dependent per-file limits and still encourages careful storage strategy.
- **Therefore: ship download scripts + checksums, not binaries.**

Model Registry Pattern (configs/model_registry.json)

```
{
  "schema_version": 1,
  "models": [
    {
      "model_id": "sam2_tiny",
      "family": "sam2",
      "backend": "onnxruntime",
      "format": "onnx",
      "artifact": {
        "uri": "https://<host>/releases/download/v0.1.0/sam2_tiny.onnx",
        "sha256": "REPLACE_WITH_SHA256",
        "size_bytes": 123456789
      },
      "license": "requires_review",
      "notes": "Downloaded on first run; cached under XDG cache dir."
    }
  ]
}
```

Download-on-First-Run + Checksum Verification

Recommended cache location:

- Default: ${XDG_CACHE_HOME:-~/.cache}/son-segserve/models/
- Override via: SON_CACHE_DIR=/path

```
import hashlib
from pathlib import Path

def sha256_file(path: Path, chunk_size: int = 1024 * 1024) -> str:
    h = hashlib.sha256()
    with path.open("rb") as f:
        for chunk in iter(lambda: f.read(chunk_size), b""):
            h.update(chunk)
    return h.hexdigest()

def verify_sha256(path: Path, expected_hex: str) -> None:
    actual = sha256_file(path)
    if actual.lower() != expected_hex.lower():
        raise ValueError(
            f"SHA256 mismatch for {path.name}: {actual} != {expected_hex}"
        )
# Principle: fail closed — do not load mismatched artefacts.
```

Artefact Distribution Options

Option	Pros	Cons / Limits	Best Fit

Versioned Release Assets	Simple UX; integrates with tags; supports large binaries; no total bandwidth/size limits; 2 GiB per asset; up to 1000 assets per release	Asset management is per-release; must maintain checksums; 2 GiB per file cap	Default for model weights + ONNX exports
OCI Container Registry	Good for containerised distribution; integrates with CI publishing	Per-layer limits; requires registry auth and careful layering	Shipping the service image; optional for 'model-as-OCI'
External Object Storage (S3/GCS)	Scales to very large artefacts; independent lifecycle	Requires account + credentials + cost + policy; increases onboarding complexity	Optional for enterprise forks; avoid in baseline

Dataset Handling Rules and Licence Flags

Introduce a dataset compliance gate: every dataset script must write a data/processed/<dataset>/MANIFEST.json containing:

- Download URLs
- File checksums (where feasible)
- Licence field (ok_for_demo, noncommercial, requires_per_item_verification, etc.)
- Citation guidance

Licence Caveats per Dataset

- **Open Images:** annotations are CC BY 4.0; images are listed as CC BY 2.0, but publishers disclaim warranties and advise verifying each image licence.
- **YouTube-VOS:** annotations under CC BY 4.0, but the dataset is released for non-commercial research only — generally incompatible with a broadly redistributable public demo intended for mixed use. Mark as 'requires review / likely exclude from default demo.'
- **VOC2012:** images sourced from Flickr; use must respect Flickr terms of use (licence is not uniformly permissive). Red flag for public demo assets committed to the repo.
- **DAVIS:** train+val is 90 sequences; evaluation uses region J and boundary F. Good for video-seg evaluation but not a 'tiny quickstart'.
- **COCO:** large-scale instance segmentation masks. Using subsets for demos is recommended to keep download weight manageable.

Conservative Recommendations for Demo Defaults

- **Default demo dataset:** Open Images masks (because mask annotations are explicit and large-scale) but with a prominent 'verify image licences' warning.
- **Default demo fixtures committed to repo:** synthetic images you generate yourself (no third-party licence risk).
- Keep YouTube-VOS and VOC off by default in scripts unless the user explicitly accepts their constraints.

Governance and Repo Lifecycle

Repository Lifecycle Flow

```
flowchart LR
  dev[Developer workstation] --> pr[Pull Request]
  pr --> ci[CI: lint + tests + security checks]
  ci -->|pass| merge[Merge to main]
  merge --> build[Build: container + docs]
  build --> release[Release: tag + release assets]
  release --> deploy[Deploy: VM / container runtime]
  deploy --> monitor[Monitor: metrics + logs + drift]
  monitor --> dev
```

Branch Protection Rules (Recommended)

- Require PR reviews before merging
- Require status checks before merging
- Require conversation resolution before merging
- Require signed commits (optional but recommended for high-integrity repos)
- Require linear history (optional; helps traceability)

Also recommended: add a CODEOWNERS file and enable 'Require review from Code Owners' to ensure sensitive areas (security, CI, release scripts) are reviewed.

Suggested Branch Naming Strategy

- main (protected)
- feature/<short-scope>
- fix/<issue-id>-<short-scope>
- release/<version> (optional, if you gate releases separately)

Recommended Security Settings

- Secret scanning — enable for public repositories.
- Dependency update automation via dependabot.yml.
- Code scanning (CodeQL default setup) — eligible for public repos with CI enabled; scans on pushes/PRs and on a schedule.

Tests, Monitoring, Security, Reproducibility

Test Suite Inventory

Keep tests small, deterministic, and licence-safe:

- **Unit tests:** pure functions (hashing, config parsing, prompt parsing, mask post-process).
- **Integration tests:** start container, hit /healthz, /segment, /metrics.
- **Model regression ('golden set'):** use generated synthetic images + fixed prompts; store expected summary statistics (mask area, connected-components count) instead of raw masks to avoid brittle exact pixel equality. Track a hash of the model artefact + inference backend version in the regression output.
- **Data validation tests:** validate dataset manifests and expected directory layouts after download (do not run full dataset downloads in CI).
- **Synthetic tests:** fuzz prompts (invalid box coords, empty points, huge images) to harden request validation.

Local Run Commands (Developer Runbook)

```
pip install -e ".[dev]"
pre-commit run -a
pytest
docker build -f docker/Dockerfile -t son-segserve:dev .
docker run --rm -p 8080:8080 son-segserve:dev
```

Monitoring and Observability Hooks

Metrics Endpoint Spec

Expose GET /metrics in Prometheus text format with Content-Type: text/plain; version=0.0.4.

Metrics to collect (minimum viable):

- HTTP request count + status
- Request latency histogram (p50/p95 derived in Prometheus/Grafana)
- Inference time histogram
- Memory RSS gauge (process metric)
- Mask count per request (gauge/hist)
- Predicted quality proxy (e.g. predicted_iou distribution)

Prometheus Scrape Config

```
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: son-segserve
    metrics_path: /metrics
    static_configs:
      - targets: ["host.docker.internal:8080"]
```

Sample Grafana Dashboard JSON Stub

```
{
  "id": null,
  "uid": "son-segserve",
  "title": "SON SegServe",
  "tags": ["segmentation", "son"],
  "timezone": "browser",
  "schemaVersion": 41,
  "panels": [
    {
      "type": "timeseries",
      "title": "HTTP latency (p95)",
      "targets": [
        {
          "expr": "histogram_quantile(0.95, sum(rate(http_request_duration_seconds_bucket[5m])) by (le))"
        }
      ],
      "gridPos": { "h": 8, "w": 24, "x": 0, "y": 0 }
    }
  ]
}
```

Security and Licensing Checks

Licence Auditing Workflow

- 1) **Model licence:** verify upstream terms (code + checkpoints) and write a model card under data/metadata/model_cards/. If any ambiguity exists, set license: requires_review and do not include weights in default releases.
- 2) **Dataset licence:** record dataset homepage, licence text, and restrictions in dataset card.
- 3) **Repo licence:** choose a clear repo licence (Apache-2.0 or MIT). Licence detection works with standard filenames.
- 4) **Enforce a 'no large files' policy:** CI check that fails if a PR adds *.pt, *.onnx, or files above a threshold.

Reproducibility Steps

- Record seeds and deterministic settings in configs/service.yaml and print them at startup.
- Add a runtime --deterministic flag: enables deterministic algorithms where possible and documents potential speed impacts.
- Pin container base images and use multi-stage Dockerfiles as per Docker best practices.
- Pin CI Actions to commit SHAs.
- For releases: include artefact checksums alongside binaries as separate *.sha256 assets.

Dockerfile (Recommended Baseline)

Multi-stage builds and .dockerignore are first-class Docker features that reduce image size.

```
# syntax=docker/dockerfile:1
FROM python:3.11-slim AS build
WORKDIR /app

RUN python -m pip install -U pip
COPY pyproject.toml README.md LICENSE /app/
COPY src/ /app/src/
RUN pip install --no-cache-dir -e ".[cpu]"

FROM python:3.11-slim AS runtime
WORKDIR /app

# Non-root user is recommended for runtime images
RUN useradd -m -u 10001 appuser
USER appuser

COPY --from=build /usr/local /usr/local
COPY --from=build /app/src /app/src
COPY configs/ /app/configs
COPY docker/entrypoint.sh /app/entrypoint.sh

EXPOSE 8080
ENV SON_CONFIG=/app/configs/service.yaml
ENTRYPOINT ["/app/entrypoint.sh"]
```

docker/entrypoint.sh

```
#!/usr/bin/env bash
set -euo pipefail
exec python -m son_segserve.api.app --host 0.0.0.0 --port 8080
```

REST API Specification (Minimum Viable)

Endpoints (versioned)

- GET /healthz → service health
- GET /readyz → readiness (model loaded, cache ok)
- GET /metrics → Prometheus metrics (text format)
- POST /v1/segment → run segmentation from image + prompt
- POST /v1/segment/video → optional: short video segmentation with temporal smoothing (advanced)

POST /v1/segment — Request JSON

```
{
  "image": {
    "content_type": "image/jpeg",
    "base64": "..."
  },
  "prompt": {
    "type": "box",
    "box_xyxy": [10, 20, 200, 220]
  },
  "options": {
    "return_format": "rle",
    "max_masks": 3,
    "stabilize": false
  }
}
```

Response JSON

```
{
  "model_id": "sam2_tiny",
  "masks": [
    {
      "mask_id": "0",
      "encoding": "rle",
      "rle": "....",
      "predicted_iou": 0.87,
      "area_px": 12345
    }
  ],
  "timing_ms": {
    "preprocess": 4.1,
    "inference": 31.7,
    "postprocess": 7.9,
    "total": 43.7
  }
}
```

CLI Commands (Thin Client)

```
son-segserve ping --api http://localhost:8080

son-segserve segment \
  --api http://localhost:8080 \
  --image ./x.jpg \
  --prompt "box:10,20,200,220" \
  --out ./mask.json

son-segserve bench \
  --api http://localhost:8080 \
  --images ./fixtures \
  --prompt "point:120,88,fg"
```

PR Checklist

Category	Checklist Item	How Reviewer Verifies
Scope	PR is single-purpose and titled clearly	Title + diff inspection
Tests	New/changed behaviour has tests	pytest passes locally/CI
CI hygiene	No new workflow secrets added without justification	.github/workflows diff
Artefact safety	No weights/datasets/binaries committed	Repo diff + CI 'large file guard'
Licensing	Dataset/model licence implications documented (if touched)	Dataset/model card updated
Reproducibility	Seeds/config updated when changing model behaviour	configs/service.yaml + logs
Observability	Metrics/logging updated for new paths	/metrics samples + logs
Docs	README/docs updated if UX changes	Docs diff + preview build

Assumptions

- Cloud provider is unspecified → recommend 'portable baseline' with a single container on a generic VM (or local).
- Hardware is unspecified → assume GPU optional; provide CPU ONNXRuntime path by default, GPU as an extra.
- Exact model size is unspecified → assume multiple model IDs (tiny/base/large) via model_registry.json.
- Licence clarity for some model checkpoints is unspecified → model licences are marked requires_review unless verified by an official artefact you explicitly approve.
- Public repo requirement implies: do not commit training data, do not commit weights unless you have explicit redistribution rights, and do not include non-commercial-only datasets in the default demo path.

Primary Sources and Dataset References

Model Primary Links

- SAM: https://segment-anything.com/
- SAM 2: https://ai.meta.com/sam2/
- MobileSAM: https://arxiv.org/abs/2306.14289

Dataset Primary Links

- COCO: https://cocodataset.org/
- PASCAL VOC: http://host.robots.ox.ac.uk/pascal/VOC/
- Open Images: https://storage.googleapis.com/openimages/web/index.html
- DAVIS: https://davischallenge.org/
- YouTube-VOS: https://youtube-vos.org/

Technical Documentation

- Prometheus configuration: https://prometheus.io/docs/prometheus/latest/configuration/configuration/
- Prometheus exposition formats: https://prometheus.io/docs/instrumenting/exposition_formats/
- Grafana dashboard JSON model: https://grafana.com/docs/grafana/latest/reference/dashboard/
- Docker best practices: https://docs.docker.com/develop/develop-images/dockerfile_best-practices/
- pre-commit: https://pre-commit.com/